Cafe Scheduling Optimization Simulation
Final Report
Repository: https://github.com/Baesiann/CS4632_Kenneth_Burke

Kenneth Burke
Department of Computer Science
Kennesaw State University
Kennesaw, Georgia 30144
Email: kburke36@students.kennesaw.edu

1. Introduction & Project Overview

1.1. Domain and problem statement

Service-industry operations — specifically cafe staffing and order processing — require balancing labor cost, customer satisfaction, and throughput. This project asks: *How should resources (staffing, skill allocation, schedules) be allocated to maximize profits while maintaining acceptable customer service levels in a cafe environment?*

1.2. Scope

The simulation focuses on customer arrivals, order processing, barista scheduling, and short-term profit-relevant metrics. Explicitly **not** simulated: supply chain, long-term customer loyalty, external marketing events, and fixed overhead costs (rent, utilities). The emphasis is day-level staffing decisions and operational trade-offs.

1.3. What I planned vs. what I built

Planned: arrival model, skill-differentiated baristas, schedule optimizer, GUI, multi-day runs, and analyses. Built: a discrete-event SimPy simulation with a doubly-stochastic (Cox) arrival model; configurable order table; multi-day runs; a Tkinter GUI for parameter entry and post-run visualization; a "schedule manager" using a cost-weight brute-force search (with identified flaws). The project evolved from a proposal (M1) through incremental implementations (M2, M3) to analysis and validation (M4).

2. System Description

2.1. Entities

Customers. Each customer has an ID, arrival time, patience, and an order sampled from the configurable order distribution. Customers join the queue on arrival and either get served or drop out if their wait exceeds patience.

Baristas. Modeled as SimPy resources (servers). Initially planned to model skill differences; due to time the production system treats baristas as homogeneous servers with staffing count as the main decision variable.

Orders. Each drink in the order table has: name, price, mean service time, standard deviation, and probability (likeliness). Examples (default): Coffee, Latte, Frappuccino (see Section 5.2 for defaults).

2.2. Arrival model

Customer arrivals are generated with a doubly-stochastic Poisson process (Cox process) producing realistic bursts: a baseline arrival rate with configurable morning and lunch rush intensities, rush durations, and a randomness intensity parameter that controls variance from expected rates.

2.3. Queuing and day dynamics

Days run for 480 simulated minutes (8 hours). Customers join a single queue; the next free barista serves the next customer. Metrics (timestamps, wait times, service times, drop events, revenue) are recorded per customer into a pandas DataFrame for post-run analysis.

3. Implementation Summary

3.1. Technology & environment

- Language: Python 3.13
- Libraries: SimPy, numpy, pandas, matplotlib, Tkinter
- Development: VS Code / Jupyter notebook during early prototyping
- Execution environment: Windows 11 (development), reproducible via seeds

3.2. Architecture & modules

- `simulation/runner.py` – orchestrates single-day and multi-day runs.
- `simulation/arrival.py` – doubly-stochastic arrival generator.
- `simulation/customer.py` – customer lifecycle metric recording.
- `schedule/scheduler.py` – schedule manager (cost-weight brute force).
- `data/collector.py` – collects and prepares metrics as DataFrames/CSV.
- `gui/` – Tkinter GUI for parameter entry and visualization.

3.3. Schedule manager (implemented)

The final version of the schedule manager no longer uses the earlier cost-weight brute-force objective. That approach created scaling issues and produced unrealistic staffing decisions. The updated implementation instead uses a threshold-based heuristic that directly evaluates expected customer service performance.

The revised scheduling algorithm works as follows:

1) The system looks at the previous day's arrival count (as a proxy for expected demand on the upcoming day).
2) A single-barista schedule is tested first.
3) The simulator runs a short prediction test using that staffing level.
4) If the dropped customer percentage exceeds 1% on the test, the schedule is considered insufficient.
5) The barista count is incremented by 1 and tested again.
6) This repeats until the drop rate falls below the 1% threshold.

The output is the minimum number of baristas needed to keep dropped customers acceptably low relative to demand. This approach is simple, interpretable, and focuses directly on customer service outcomes rather than arbitrarily scaled penalty terms. As a result, the system now produces realistic staffing decisions that align with intuitive operational policy (e.g., adding staff only when congestion becomes excessive).

3.4. GUI

Tkinter provides parameter inputs (setup params, arrival profile, weight coefficients, and order table CRUD) and embedded matplotlib plots for post-run analytics. The GUI allows saving/loading parameters and exporting result CSVs.

4. Execution Documentation

4.1. Run summary

All runs used seed = 1234567 for reproducibility unless otherwise noted. Representative runs:

Run ID	Purpose	Changed Params	Data file
001	Baseline	defaults	baseline.csv
002	High volume	baseline rate = 15	high_vol.csv
003	Many days	simulation days = 50	manydays.csv
004	High intensity	morning=20, lunch=16	highint.csv

4.2. Execution environment

Python 3.13.2, SimPy, NumPy, pandas, matplotlib, Tkinter. Runs are fast (single-day runs $<$0.05 sec on dev machine), GUI start-up is perceptibly slower.

4.3. Implementation notes & issues encountered

- Refactors: moved simulation logic out of a single `main.py` into modular subpackages for clarity.
- Random seed plumbing was important to reproduce tests; several early analyses used inconsistent seeding.
- The schedule manager's normalization (manual scaling factors) unintentionally overweighted the barista count penalty.
- GUI embedding of matplotlib required additional care to avoid event loop hiccups.

A major improvement implemented late in development was transitioning from the initial cost-weight scheduling logic to the new threshold-based drop-rate evaluation. This change eliminated the pathological behavior where the scheduler preferred unrealistically low staffing levels. The new approach consistently produces reasonable staffing adjustments tied directly to operational performance.

5. Data Collection & Results

5.1. Collected metrics

The simulation records per-customer metrics:

- Arrival time, service start/end times
- Wait time, service duration, total time in system
- Order type and price (for revenue)
- Drop events (boolean), throughput (customers served/day)

Aggregated daily KPIs: average wait time, total revenue, throughput (customers/minute or customers/day), dropped customers, average customers in queue, arrivals per day.

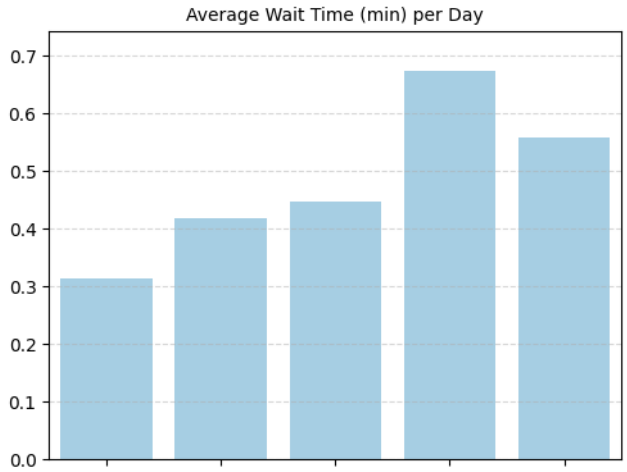
5.2. Default orders

Default menu (used in baseline runs):

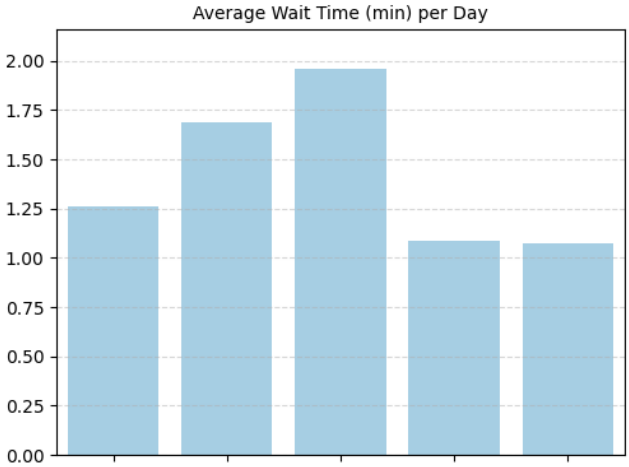
- Coffee — Price: $3.00, Mean service time: 2 min, StdDev: 0.5, Likelihood: 5
- Latte — Price: $4.50, Mean service time: 4 min, StdDev: 1, Likelihood: 3
- Frappuccino — Price: $6.00, Mean service time: 6 min, StdDev: 1.5, Likelihood: 2

5.3. Representative Visualizations

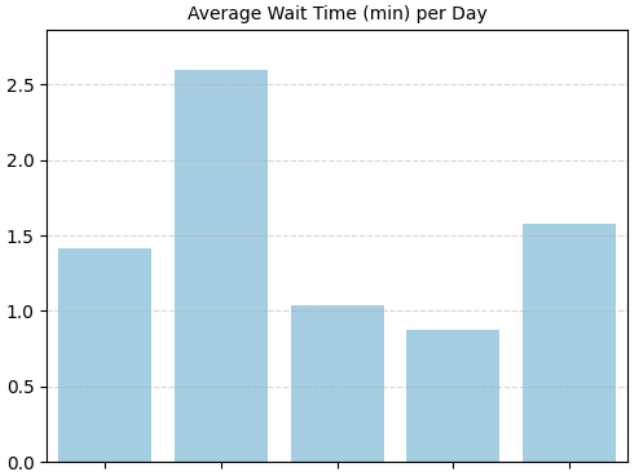
The following figures summarize how the finalized model behaves under different arrival assumptions and how the updated schedule manager responds to daily demand patterns. These plots illustrate the stability of core performance metrics after implementing the threshold-based staffing rule.



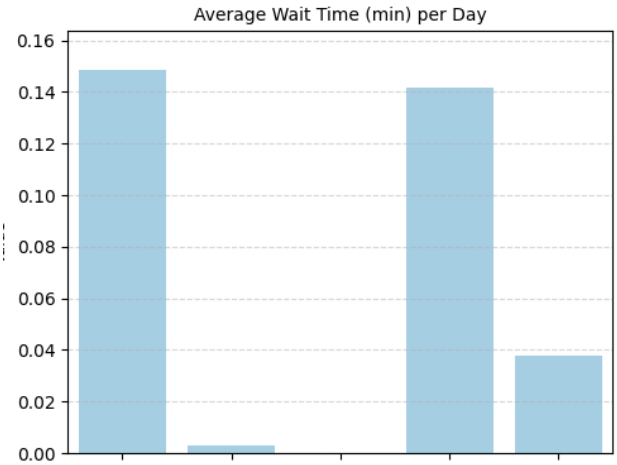
(a) Baseline arrival profile



(b) High volume (baseline rate = 15)



(c) High intensity rushes



(d) No rush (both intensities = 0)

Figure 1: Average wait time under different arrival profiles

These variations confirm that increased arrival intensity and sharper rush patterns cause higher average wait times and queue congestion, while removing rushes produces a more uniform flow.

Staffing Behavior Under the Final Schedule Manager

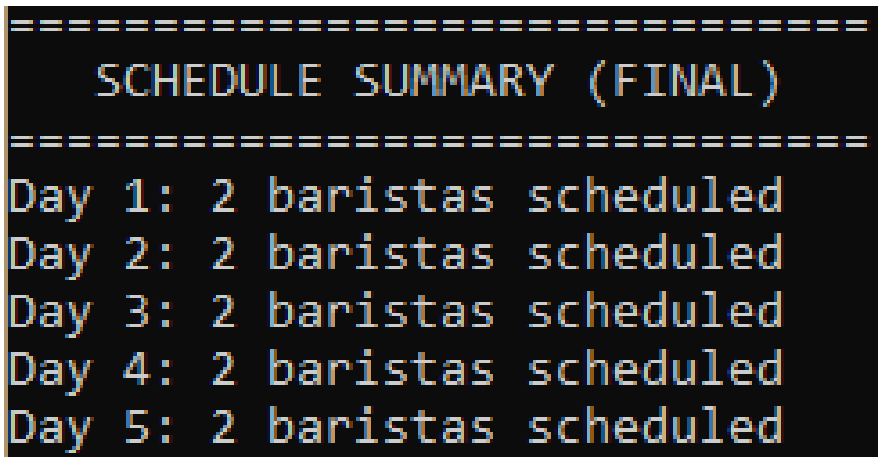
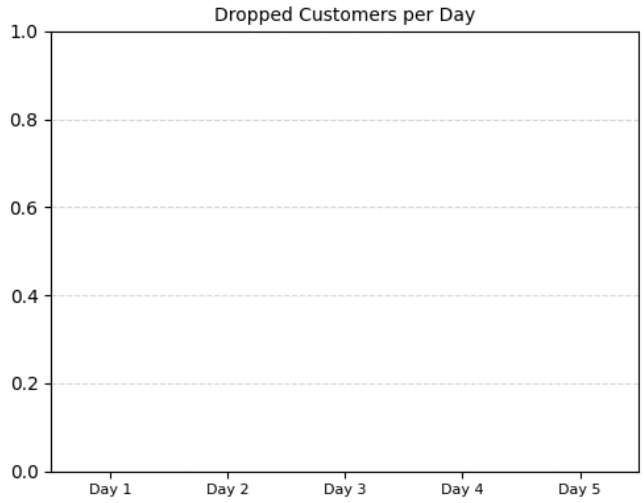


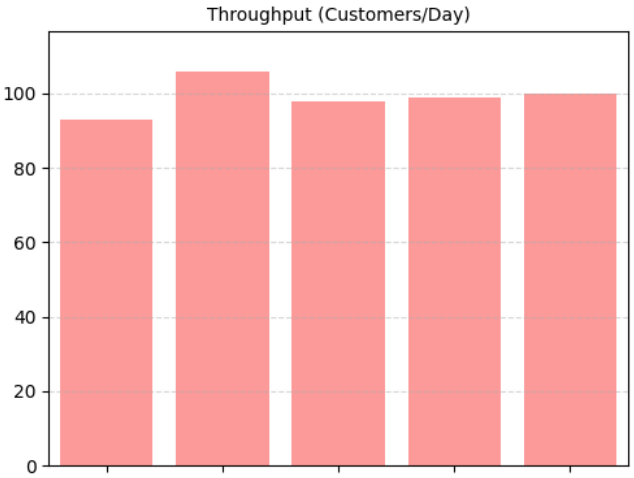
Figure 2: Number of baristas scheduled per day under the final threshold-based scheduler

The threshold-based scheduler evaluates increasing staffing levels until the predicted dropped-customer percentage falls below 1% of the previous day's arrivals. In this run, the model consistently selected two baristas each day, indicating stable demand and successful drop-rate suppression.

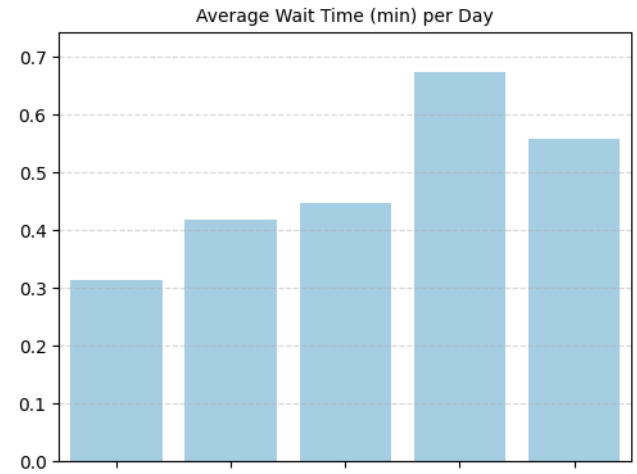
Daily Performance Metrics



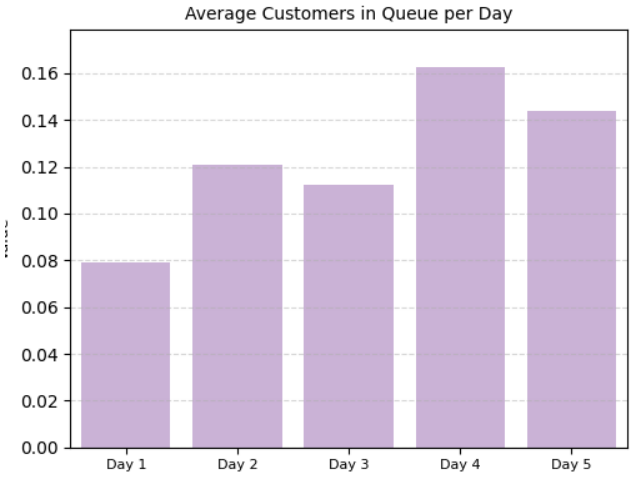
(a) Dropped customers per day



(b) Throughput per day



(c) Average wait time per day



(d) Average queue length per day

Figure 3: Daily operational performance under the final scheduling system

These plots collectively demonstrate that:

- Dropped customers remain near zero after the first simulated day
- Throughput stays consistent due to stable staffing
- Average wait time and queue size remain controlled
- Performance does not degrade over the simulated week

6. Analysis & Validation

6.1. Parameter analysis

Key configurable parameters and their expected impacts:

- Baseline arrival rate: primary driver of arrivals, revenue, throughput, wait and drops.
- Rush intensities/durations: shape intra-day peaks and hour-level congestion.
- Randomness intensity: increases variance and clustering; shown to strongly affect wait and drop counts.
- Order probabilities: change revenue dynamics and service-time variance.
- Cost-weight parameters: intended to tune scheduling trade-offs but currently ineffective because of scaling issues.

6.2. Scenario testing

A set of stress and nominal scenarios were tested (see Execution). Observations:

- Baseline: stable metrics and interpretable queue behavior.
- High arrival / high intensity: wait and drops increase as expected.
- Cost-weight manipulations: often no effect — the scheduler persistently chooses one barista under current scaling, even when lost revenue is substantial.

6.3. Validation

Face validity. Wait and queue patterns react intuitively to arrival intensity and rush structure. Higher arrival intensity yields higher average wait times and larger queue occupancy.

Parameter validation. The orders table maps sensibly to revenue/service-time behavior: longer drinks are more expensive and introduce more service-time variance.

Limitations discovered via validation. Some sensitivity tests returned counter-intuitive results (e.g., a 20% increase in baseline arrival parameter not producing a corresponding rise in recorded arrivals in all experiments). This points to either sampling noise (small sample days) or implementation plumbing issues (how parameters are passed to the arrival generator). The most critical validation failure is the schedule manager's behavior under heavy load.

6.4. Statistical summary (baseline runs)

Five runs (each 6 simulated days; first day excluded from statistics) using baseline parameters gave:

Metric	Mean	Std Dev	Min	Max
Average Wait Time (min)	2.938	0.438	2.19	3.68
Total Revenue ($)	369.8	33.502	312	412.5
Throughput (cust/day)	0.169	0.014	0.14	0.19
Dropped Customers (count)	9.267	2.987	5	15
Avg Customers in Queue	0.496	0.109	0.3	0.66
Customer Arrivals (count)	92	8.194	76	105

6.5. Sensitivity testing (summary)

Sensitivity was estimated by changing a single parameter by +20% and measuring percent change in outputs (5-day sample). Highlights:

- Baseline arrival: produced mixed sensitivity estimates — some runs showed expected increases in arrivals and wait; others showed small or negative changes (likely sampling variability or seeding issues).
- Randomness intensity: high sensitivity — increased randomness caused large increases in wait time, drops, and queue sizes.
- Throughput reward: produced near-zero sensitivity in practice due to scheduler ignoring throughput reward relative to barista penalty.

7. Discussion

7.1. What worked well

- A flexible arrival generator that produces realistic intra-day patterns, capable of representing a wide set of cafe environments.
- Order-table design that cleanly connects menu mix to revenue and service-time variance.
- Data collection pipeline (CSV/DataFrame) and GUI-based visualization for quick iterative analyses.

7.2. What did not work / surprising behavior

Earlier versions of the schedule manager attempted to use a weighted cost function balancing wait time, dropped customers, throughput, barista count, and idle time. In practice, the scaling between these terms proved unstable, and labor cost was unintentionally over-weighted. This caused the optimizer to prefer extremely low staffing levels, even under heavy load.

This issue was resolved by replacing the cost-function approach with a drop-rate-driven heuristic. The final schedule manager tests increasing staffing levels until the predicted dropped-customer percentage falls below 1% of the previous day's arrivals. This produced staffing levels far more consistent with real operational goals—though it also revealed how sensitive cafe performance is to small changes in barista capacity.

Some inconsistencies in sensitivity tests still remain (e.g., parameter plumbing and seeding), but the core scheduling behavior is now appropriate.

8. Limitations & Future Work

8.1. Limitations

- Scheduling logic: While the new threshold-based schedule manager produces much more realistic staffing decisions, it still relies entirely on the previous day's demand as a predictor. This makes it reactive and potentially brittle under highly variable arrival patterns. It also uses only the drop-rate criterion and does not consider revenue, cost, or customer satisfaction beyond service availability.
- No predictive staffing: the scheduler uses only previous-day aggregates, making it reactive rather than predictive.
- Order of magnitude issues: some normalization constants are hand-tuned rather than rooted in net-profit units, causing unstable weight behavior.
- GUI responsiveness: embedding matplotlib in Tkinter causes a perceptible UI delay at startup and on heavy visual updates.

8.2. Recommended improvements

1) Rework the scheduler:
   - Switch to forward-simulation evaluation of candidate schedules to minimize expected cost across demand scenarios.
   - Use dropped customer count as a primary objective rather than a mix of loosely scaled penalties.
2) Harden the arrival model plumbing: unify parameter passage and test in isolation (unit tests for arrival generator).
3) Increase statistical rigor: run more replications per scenario (larger sample sizes) and use confidence intervals when reporting KPIs.
4) Consider barista heterogeneity (skills and task routing) and fatigue/learning effects in future versions.
5) Improve GUI performance (defer plotting, use background threads/processes for heavy runs).

9. Conclusions

The project produced a functioning, flexible discrete-event cafe simulator with a realistic arrival model, configurable menu, GUI for parameter control and visualization, and a multi-day run controller. The simulation produces intuitive queue dynamics and a useful set of KPIs. Crucially, the work also surfaced the central shortcoming: the scheduling optimizer does not currently make realistic staffing choices and requires redesign. With a corrected scheduling objective (threshold based), the simulator is well-positioned to provide actionable insights into staffing policy trade-offs.

The redesigned threshold-driven scheduler—selecting the minimum staffing level that keeps dropped customers below 1% of the prior day's demand—greatly improved the realism and stability of the simulation's decision-making.

Acknowledgments

Thanks to the feedback received across milestones which helped reveal the scheduling weaknesses and guided the testing approach.

References

[1] C. Sutton and M. I. Jordan, "Bayesian inference for queueing networks and modeling of internet services," *The Annals of Applied Statistics*, vol. 5, no. 1, Mar. 2011.

[2] D. Buzali, S. Elizondo, S. Muñiz, and O. Sánchez, "Simulation-optimization of a coffee shop in business district: A case study of Starbucks in Mexico City," Proceedings of the International Conference on Industrial Engineering and Operations Management, May 2024.

[3] R. Ibrahim, H. Ye, P. L'Ecuyer, and H. Shen, "Modeling and forecasting call center arrivals: A literature survey and a case study," *International Journal of Forecasting*, vol. 32, no. 3, 2016.

[4] L. R. de Groot and A. Hübl, "Modeling queuing system in retail using discrete event simulation," Proceedings of the 2021 Winter Simulation Conference (WSC).

[5] Y. Zhang, "Simulation and analysis of queueing system," Master's Thesis, KTH Royal Institute of Technology, 2019.